PTO 05-[4430]

Japanese Patent

WO 99/21092

FILE PROCESSING METHOD, DATA PROCESSING DEVICE, AND STORAGE MEDIUM

[Fairu Shori Hoho, Deta Shori Sochi Oyobi Kioku Baitai]

Masahiro Kataoka and Takashi Tsubokura

UNITED STATES PATENT AND TRADEMARK OFFICE
Washington, D.C. June 2005

Translated by: Schreiber Translations, Inc.

2

Country : Japan

Document No. : WO 99/21092

Document Type : PCT

Language : Japanese

Inventor : Masahiro Kataoka and Takashi Tsubokura

Applicant : Fujitsu Ltd.

IPC : G 06 F 12/00

Application Date : October 20, 1998

Publication Date : April 29, 1999

Foreign Language Title : Fairu Shori Hoho, Deta Shori Sochi Oyobi Kioku Baitai

English Title : FILE PROCESSING METHOD, DATA PROCESSING DEVICE, AND STORAGE MEDIUM

/1[1]

Specification

FILE PROCESSING METHOD, DATA PROCESSING DEVICE, AND STORAGE MEDIUM

Technical field

The present invention pertains to a file processing method, a data processing device, and a storage medium. In particular, the present invention pertains to a fill processing method that compresses a file such as dictionary file regarding one or several dictionaries and encyclopedia, stores it in a storage medium, and reads it out, a data processing device, and a storage medium for storing a film such as compressed dictionary file.

Recently, information such as dictionary and encyclopedia has been stored in advance in a recording medium such as CD-RO, and the information such as dictionary and encyclopedia has been read out and displayed by accessing the CR-ROM by a computer. Thus, an enormous information such as dictionary and encyclopedia can be stored in one sheet of very compact CD-ROM. Also, since a necessary information can be read out of the CD-

---

[1] Numbers in the margin indicate pagination in the foreign text.

ROM instead of obtaining the necessary information by opening the dictionary, encyclopedia, etc., during the use of the computer, the labor and time for obtaining the necessary information is largely reduced.

Background technique

In the conventional CD-ROM in which information such as dictionary and encyclopedia is stored, a dictionary file consists of a dictionary data and an index data (hereinafter, called an index data). For example, in an encyclopedia, the dictionary data includes an image data which shows an animal if words indicate the animal (hereinafter, called an image data), a voice data which shows chirping of a bird if words indicate the bird (hereinafter, called a voice data), etc. An index is used to search for a desired dictionary data from the dictionary file, installed for the dictionary data, and also called a /2
keyword. Header pointer, item pointer, etc., are included in an index data. A header word is included in a header data. Also, header word, comment sentence, etc., are included in an item data.

Since the storage capacity of a conventional CD-ROM is relatively large, text data and index data are stored in the CD-ROM without being compressed. On the other hand, the amount of information of image data and voice data, especially image data is large, and they are respectively compressed by an appropriate compression method and stored in the CD-ROM.

However, if one sheet of CD-ROM is required for each dictionary or encyclopedia, the usage conditions of dictionary data are poor. Accordingly, it is desirable to store information of several dictionaries and encyclopedias in one sheets of CD-ROM, and in this case, there is a possibility that the amount of information to be stored exceeds the storage capacity of one sheet of CD-ROM, even if the dictionary data are compressed. Also, even if the dictionary file to be stored in the CD-ROM is a single dictionary or encyclopedia, if the amount of information of the dictionary file is increased, there is a possibility that it exceeds the storage capacity of one sheet of CD-ROM, even if the dictionary data is compressed.

For this reason, it is considered that the entire dictionary file containing the index data as well as the dictionary data is compressed and stored in the CD-ROM. However, a method that compresses the entire dictionary file with good efficiency by a relative simple method and can expand the compressed dictionary file in a short time is not proposed. Especially, for the dictionary and the encyclopedia, since the amount of information of the index data is large, if time is required for a processing for restoring the index data during the expansion of the dictionary file, the access time to a desired index data or dictionary data is lengthened, so that the usage conditions of the dictionary or encyclopedia are deteriorated.

Also, for example, in case the dictionary data are compressed at an item unit or fixed length unit of the indexes, especially in the dictionary or encyclopedia, since the amount of information of the index data is large, time is required /3 for the expansion procession of a dictionary file similarly to the above-mentioned case, so that the usage conditions of the dictionary or encyclopedia are deteriorated. For example, in Japanese Kokai Patent Application No. Hei 9[1997]-26969, a telephone book search system using a method similar to it is proposed, however in the proposed method, the index data are not compressed. The reason for this is that in the telephone book, the amount of information of the index data is smaller than the amount of information of telephone numbers, names, names of corporations, and addresses and the information compression efficiency as a whole is not considerably improved by compressing. For this reason, even if this proposed method is applied in storing the information of dictionary, encyclopedia, etc., into a storage medium, the information compression efficiency as the entire dictionary file is not considerably improved.

Therefore, if the amount of information of index data was relatively larger than the amount of information of dictionary data, like dictionary, encyclopedia, etc., a dictionary file could not be compressed with good efficiency and stored in a storage medium and cannot access the compressed dictionary file in a short time by a relatively simple processing.

Presentation of the invention

Accordingly, the purpose of the present invention is to provide a file processing method, which compresses a file such as dictionary file with good efficiency, even if the amount of information of an index data is relatively larger than the amount of information of a dictionary data, stores it in a recording medium, and has access to the compressed file such as compressed dictionary file in a short time by a relatively simple processing, a data processing device, and a storage medium.

The purpose of the present invention is to provide a file processing method consisting of a step that divides a data and an index data for said data into several sections and attains a compressed file by compressing and a step that stores said compressed file and the address information after the compression of the above-mentioned sections in a storage medium. According to the present invention, a file such as dictionary file being constituted by data such as index and texts of each/4 item can be compressed with good efficiency and stored in the storage medium, and if the compressed file is expanded for each section, the file can be searched at high speed by a relatively simple processing.

If the above-mentioned have a fixed length, it is necessary to include the address information before compressing in the compressed file, and the data compression efficiency can be improved. If the above-mentioned sections have a variable length and the above-mentioned storage step also stores the address information before compressing in the above-mentioned storage medium, the data can be expanded at high speed by setting appropriate sections in accordance with the kind of data and the classification.

If a step that reads the above-mentioned compressed file out of the above-mentioned storage medium for each of the above-mentioned sections, expands it, and stores the restored data and index data in an auxiliary storage device is further included, the auxiliary storage device that enables a high-sped data access is used, so that the file search speed can be improved.

At the above-mentioned compression step, if a compression algorithm and a compression parameter common to the data and the index data for each section are used, the data compression processing and the data expansion processing during the data expansion can be simplified, so that Huffman code, universal code, etc., can be used as a specific compression algorithm.

Furthermore, another purpose of the present invention is to provide a file processing method consisting of a step that divides a data and an index data for said data into several sections and reads a compressed file, which is attained by compressing, along with the address information after the compression of the above-mentioned sections out of a storage medium for each of the above-mentioned sections and a step that expands said compressed file and restores the data and the index data. According to the present invention, a compressed file such as compressed dictionary file is expanded for each section, so that a file can be searched at high speed by a relatively simple processing.

Another purpose of the present invention is to provide a file processing device equipped with a means that divides a data and an index data for said data into several sections and attains a compressed file by compressing and a means that stores said compressed file and the address information after the compression of the above-mentioned sections in a storage medium. According to the present invention, a file being constituted by data such as index and texts of each item can be compressed with good efficiency and stored in the storage medium, and if the compressed file is expanded for each section, the file can be searched at high speed by a relatively simple processing.

Furthermore, another purpose of the present invention is to to provide a file processing device consisting of a means that divides a data and an index data for said data into several sections and reads a compressed file, which is attained by compressing, along with the address information after the compression of the above-mentioned sections out of a storage medium for each of the above-mentioned sections and a means that expands said compressed file and restores the data and the index data. According to the present invention, a compressed file is expanded for each section, so that a file can be searched at high speed by a relatively simple processing.

Another purpose of the present invention is to provide a storage medium that stores the information readable by a computer. The storage medium stores a program, which is equipped with a means that divides a data and an index data for said data into several sections and reads a compressed file, which is attained by compressing, along with the address information after the compression of the above-mentioned sections out of the above-mentioned storage medium for each of the above-mentioned sections and a means that expands said compressed file and restores the data and the index data, in the computer. According to the present invention, a compressed file is expanded for each section, so that a file can be searched at high speed by a relatively simple processing.

Furthermore, another purpose of the present invention is to provide a storage medium that stores the information readable by a computer. In the storage medium, a data and an index data for said data are divided into several sections, and a compressed /6 file attained by compressing is stored along with the address information after the compression of the above-mentioned sections in a storage medium. Said compressed file is attained by compressing using a compression algorithm and a compression parameter common to the data and the index data for each section. According to the present invention, a file can be compressed with good efficiency and stored in the storage medium, and if the compressed file is expanded for each section, the file can be searched at high speed by a relatively simple processing.

The purpose of the present invention is to provide a storage medium that stores the information readable by a computer. The storage medium stores a program, which implements a sequence that divides a dictionary data and an index data for said dictionary data into several sections and attains a compressed dictionary file by compressing and a sequence that stores said compressed dictionary file and stores the storage medium, in the computer. According to the present invention, a file can be searched at high speed by a relatively simple processing.

Furthermore, another purpose of the present invention is to provide a storage medium that can read a compressed file by a computer. The storage medium can read the stored compressed file having a compressed data area where a data and an index data for said data are divided into several sections and a compressed data is recorded, a storage address information area where the address information after the compression of the above-mentioned section is stored, and a compression parameter area where the compression parameter used in the above-mentioned compression processing is stored. According to the present invention, a file can be searched at high speed by a relatively simple processing.

Therefore, according to the present invention, even if the amount of information of index data is relatively larger than the amount of information of dictionary data, like dictionary, encyclopedia, etc., a file such as dictionary file can be compressed with good efficiency and stored in a storage medium and can access the file such as compressed dictionary file in a short time by a relatively simple processing.

Furthermore, other purposes and merits of the present /7 invention will be clarified by the following explanation along with the figures.

Brief description of the figures

Figure 1 is a block diagram showing an outlined constitution of a computer system that adopts an application example of the file processing method.

Figure 2 a flow chart for explaining a compression parameter arithmetic processing of a CPU.

Figure 3 shows the data structure of a compression parameter.

Figure 4 is a flow chart for explaining a data compression processing and an address information arithmetic processing of the CPU.

Figure 5 is a flow chart for explaining a compression file synthesis processing and a compression file storage processing of the CPU.

Figure 6 explains the synthesis of a compressed file.

Figure 7 is a flow chart for explaining an index reading processing of the CPU.

Figure 8 is a flow chart for explaining a data expansion processing of the CPU.

Figure 9 is a flow chart for explaining a text reading processing of the CPU.

Preferred embodiment of the invention

An application of the file processing method and the data processing device of the present invention is explained. In the application example of the file processing method and the data processing device, an application example of the storage medium of the present invention is used. Also, in the application example of the storage medium, the present invention is applied to a CD-ROM, however the recording medium itself is not limited to the CD-ROM. Needless to say, the present invention can also be similarly applied to optical information recording media other than the CD-ROM, photomagnetic recording media such as photomagnetic disk, magnetic storage media such as floppy disk, and various kinds of semiconductor memory devices.

Figure 1 is a block diagram showing an outlined /8 constitution of a computer system that adopts an application example of the file processing method and corresponds to the application example of the data processing device. In the figure, the computer system is roughly constituted by a central processing unit (CPU) 1 connected by a bus 9, a main storage device 2 consisting of random access memory (RAM), etc., an auxiliary storage device 3 consisting of hard disk drive, etc., an input device 4 consisting of keyboard, mouse, etc., a display device 5, and a CD-ROM input and output device 6 consisting of CD-ROM drive, etc. Each element itself constituting the computer system can be constituted by a well-known constitution.

The input device 4 is used in inputting instructions and data into the CPU 1 and carries out a desired processing of a user by implementing a program stored in the auxiliary storage device 3 based on these instruction and data. The program stored in the auxiliary storage device 3 may be installed in advance or may also be loaded from a CD-ROM 6a loaded into the CD-ROM input and output device 6. The main storage device 2 is used in temporarily storing intermediate results of arithmetic processing, etc., and data being used in the arithmetic of the CPU 1. The display device 5 displays the result of the arithmetic of the CPU 1 and a message for stimulating the user to input the instruction and the data. Also, in stead of the display device 5 or in addition to the display device 5, a printer (not shown in the figure) for printing the results of the processing of the CPU 1 may also be connected to the bus 9.

First, a file storage processing that stores a dictionary file such as dictionary and encyclopedia in the CD-ROM 6a loaded into the CD-ROM input and output device 6 is explained. The file storage processing is roughly constituted by a data compression processing of index, text, etc., an address information arithmetic processing, a compression file synthetic processing, and a compressed file storage processing. In this application example, for convenience of explanation, in the CD-ROM 6a, a program for implementing a file storing processing on the CPU 1 is stored, and the CPU 1 reads the program out of the CD-ROM 6 by a well-known method and loads it into the /9 auxiliary storage device 3. Also, the dictionary file such as dictionary and encyclopedia is transferred from a host device (not shown in the figure) and stored in the auxiliary storage device via the bus 9, or the dictionary file is read out of a CD-ROM different from the CD-ROM input and output device.

1a) Compression parameter arithmetic processing:

Figure 2 is a flow chart for explaining a compression parameter arithmetic processing of the CPU 1. In the figure, a step S1, the auxiliary storage device 3 is accessed, and a dictionary file is opened. At step S2, one character, that is, for example, a 16-bit code is read out of the dictionary file. At step S3, the appearance frequency of the 16-bit code read is counted by an appearance frequency counter in the CPU 1. At step S4, whether or not the final character of the dictionary file is processed is decided, and if the decision result is NO, the processing proceeds to step S2.

On the other hand, if the decision result at step S4 is YES, the dictionary file is closed at step S5. At step S6, the 16-bit code is sorted in the appearance frequency sequence, and at step S7, for example, 1024 pieces of 16-bits codes are selected in the appearance frequency sequence. At step S8, the unselected remaining 16-bit codes are decomposed into 8-bit codes, and the appearance frequency of the 8-bit codes is calculated. At step S9, the appearance frequency of the 8-bit codes is set to about 1/2, and the appearance frequency of the 16-code bits is corrected.

At step S10, a save file of a compression parameter is opened in the auxiliary storage device 3. At step S11, 1024 pieces of 16-bit codes and their appearance frequency are written into the save file of the compression parameter. Also, at step S12, 256 pieces of 8-bit codes and their appearance frequency are written into the save file of the compression parameter. At step S13, the save file of the compression parameter is closed, and the processing is finished.

Figure 3 shows the data structure of the compression parameter. In the compression of Huffman codes, as shown in the figure, the compression parameter consisting of 256 kinds of appearance frequencies for each of 1024 kinds of 16-bit codes and 256 kinds of appearance frequencies for each of 8-bit codes. These appearance frequencies become data for generating a Huffman tree. Also, in the compression of universal codes, the compression parameter is constituted by a try tree or a registration symbol example for generating it and its reference number.

1b) Data compression processing and address information arithmetic processing:

Figure 4 is a flow chart for explaining a data compressing processing and an address information arithmetic processing of the CPU 1. In the figure, at step S21, from the appearance frequency of 8-bit codes and the appearance frequency of 16-bit codes, a conversion table , that is, the Huffman tree is prepared since the Huffman compression is carried out in this application example. At step S22, the dictionary file in the auxiliary storage device 3 is opened. At step S23, the save file of the compressed data and the save file of the address information are opened in the auxiliary storage device 3.

At step S24, one section is read out of the dictionary file. This section may be a fixed length or a variable length, and in this application example, the fixed length is used for convenience of explanation. Also, the above-mentioned section is called a block. At step S25, the compressed data of one section is calculated using the Huffman tree. At step S26, an end code is added to the end of one section. Also, at step S27, the compressed data is written into the save file of the compressed data.

At step S28, the address information in which said section is stored is calculated. For example, in case the section has a fixed length, the address information is calculated based on the section number given to each section. At step S29, the address information is written into the save file of the address information. At step S30, whether or not the final section is processed is decided, and if the decision result is NO, the processing returns to step S24. Whether or not the final /11 section is processed can be decided based on the section number or the final section code added to the final section.

On the other hand, if the decision result at step S30 is YES, the save file of the compressed data and the save file of the address information are respectively closed at step S31. Also, at step S32, the dictionary file is closed, and the processing is finished.

1c) Compressed file synthesis processing and compressed file storage processing:

Figure 5 is a flow chart for explaining a compressed file synthesis processing and a compressed file storage processing of the CPU 1. In the figure, at step S41, the compressed file is opened in the auxiliary storage device 3. At step S42, the save file of the compression parameter in the auxiliary storage device is opened, and at step S43, the compression parameter of the save file of the compression parameter is copied to the compressed file. At step S44, the save file of the compression parameter is closed.

At step S45, the save file of the address information in the auxiliary storage device 3 is opened, and at step S46, the address information of the save file of the address information is copied to the compressed file. At step S47, the save file of the address information is closed. Furthermore, at step S48, the save file of the compressed data in the auxiliary storage device 3 is opened, and at step S49, the compressed data of the save file of the compressed data is copied to the compressed file. At step S50, the save file of the compressed data is closed. At step S51, the compressed file is stored in the CD-ROM 6a by the CD-ROM input and output device 6. Also, at step S52, the compressed file is closed, and the processing is finished.

Figure 6 explains the synthesis of a compressed file of 1a) compression parameter arithmetic processing, 1b) data compression processing and address information arithmetic processing, and 1c) compressed file of compressed file synthesis processing and compressed file storage processing as mentioned above. In the figure, (a) shows a compression parameter and is a compression parameter for the compression of the Huffman /12 codes in this application example. In the figure, (b) shows a section of the dictionary file, and each section in this application example is 2 kbytes, for instance. Each section consists of a dictionary data and an index data. For example, in an encyclopedia, the dictionary includes a text data of a text for explaining the meaning of words, an image data showing an animal if the words means the animal, a voice data showing chirping of a bird if the words means the bird, etc. The index is used to search for a desired dictionary data from a dictionary file, installed for a dictionary data, and also called a keyword. The index data includes header pointer, item pointer, etc. The header data includes a header word. Also, the item data includes header word, comment sentence, etc.

In Figure 6, (c) shows a compressed data and shows a state in which each section is compressed at a fixed length or a variable length. Also, in the figure, (d) shows an address information calculated for each section, and (e) shows a compressed file being obtained by synthesizing the compression parameter, the address information, and the compressed data and adding a management information to the head. The management information includes information such as name of the dictionary file, kind of dictionary file, and kind of compression of the dictionary file being used in searching for the compressed file.

Next, a file search processing that reads the compressed file stored in the CD-ROM 6a loaded into the CD-ROM input and output device 6 and searches for a desired data is explained. The file search processing roughly consists of an index reading processing and a text reading processing and is carried out by calling a data expansion processing. In this application example, for convenience of explanation, a program for implementing the file search processing on the CPU 1 is stored in the CD-ROM 6, and the CPU 1 reads the program and the compressed file out of the CD-ROM 6a by a well-known method and loads them into the auxiliary storage device 3.

2a). Index reading processing: /13

Figure 7 is a flow chart for explaining an index reading processing of the CPU 1. In the figure, at step S61, an address information of the highest index is set based on the index data input from the input device 4 by a user. At step S62, the expansion processing is called, and a routine for implementing the expansion processing from a program for the file search processing in the auxiliary storage device 3 is read out, so that the address of the highest index in the compressed file is expanded. At step S63, the address of the upper index as the first letter of the highest index is acquired based on the above-mentioned index data. At step S64, the expansion processing is called, and the address of the upper index in the compressed file is expanded. At step S65, the address of the lower index of the next hierarchy is acquired based on the above-mentioned index data. At step S66, the expansion processing is called, and the address of the lower index of the above-mentioned next hierarchy in the compressed file is expanded. At step S67, whether or not the expansion of the address of the lowest index is finished is decided, and if the decision result is NO, the processing returns to step S65. On the other hand, if the decision result at step S67 is YES, the processing is finished.

2b) Data expansion processing:

Figure 8 is a flow chart for explaining a data expansion processing of the CPU 1. The data expansion processing is called by the index reading processing and the text reading processing. In the figure, at step S71, the user stores required expansion address, data size, and storage area in the auxiliary storage device 3, and the storage area of a sufficient size for the expanded data size is prepared in the auxiliary storage device 3. At step S72, whether or not the compressed file read out of the CD-ROM 6a and loaded into the auxiliary storage device 3 is opened is decided. If the decision result is NO, the compressed file in the auxiliary storage device 3 is opened at step S73. At step S74, the compression parameter /14 is read out of the compressed file, and the appearance frequency of 8-bit codes I the compression parameter, 16-bit codes, and its appearance frequency are read. At step S75, a Huffman tree is prepared based on the appearance frequency of 8-bit codes and the appearance frequency of 16-bit codes, and the processing advances to step S76 that will be mentioned later. Also, a flag for decision of 8-bit codes or 16-bit codes is added to the data of leaves of the Huffman tree.

If the decision result at step S72 is YES or after step S75, the address information corresponding to the expansion address required at step S76 is read out of the compressed file. At step S77, the section of the compressed data corresponding to the compressed file is read based on the address information. At step S78, the section of the compressed data is expanded from the Huffman tree, and at step S79, the expanded data is copied in the above-mentioned storage area based on the flag for decision of 8-bit codes or 16-bit codes. Also, at step S80, whether or not the expansion of the data size required for the compressed file is completed is decided.

If the decision result at step S80 is NO, the address information corresponding to the expansion address of the next section is read out of the compressed file. At step S81, the section of the compressed data corresponding to the compressed file is read based on the address information corresponding to the expansion address of the next section, and the processing returns to step S78. On the other hand, if the decision result at step S80 is YES, the processing is finished.

2c) Text reading processing:

Figure 9 is a flow chart for explaining a text reading processing of the CPU 1. In the figure, at step S91, the user counts items matched with the index in the expanded data based on the index data input from the input device 4. At step S92, the value of the item pointer of the index is set to an address on the basis of the input index data. At step S93, the expansion processing is called, and a routine for the expansion processing from a program for the file search processing in the auxiliary storage device 3 is read out, so that the text shown by the item pointer in the compressed file, that is, the dictionary data is expanded by one section.

At step S94, whether or not the dictionary data shown by the item pointer is finished is decided, and if the decision result is NO, the address of the next one section is set at step S95. Also, at step S96, the expansion processing is called, the dictionary data shown by the item pointer in the compressed file is expanded by the next one section, and the processing returns to step S94. On the other hand, if the decision result at step S94 is YES, whether or not the processing for all the items is finished is decided on the basis of the input index data at step S97, and if the decision result is NO, the processing returns to step S92. If the decision result at step S97 is YES, the dictionary data expanded for all the items is displayed on the display device 5 at step S98, and the processing is finished.

Also, the step S98 may be implemented before the step S97. In this case, at step S98, each dictionary data expanded for each item is displayed on the display device 5.

In the above-mentioned application example, for convenience of explanation the sections have had a fixed length. In this case, the data compression efficiency is good, and the address information can be restored from the compressed file, even if the address information before the compression of the sections is not stored in the compressed file. The reason for this is that the sections have a fixed length and the section numbers are given to each section, and the relative position for the other sections of each section can be calculated.

On the other hand, if the above-mentioned sections have a variable length, the data expansion rate can be improved. The reason for this is that the sections with an appropriate length can be set in accordance with the kind of data and the classification and an extra data is not required to be expanded. Also, if the sections have a variable length, it is necessary to store the address information before the compression of the sections in the compressed file. Therefore, whether the sections are set to a fixed length or a variable length may /16 be determined by whether the data compression rate has priority or the data expansion rate has priority.

Also, the dictionary file being stored in the CD-ROM 16 may be one or more. Even if several dictionary files regarding several dictionaries or encyclopedias are stored in the CD-ROM

6a, the dictionary file to be searched from the name of the dictionary file and the kind of dictionary file in the management information shown in (e) can be specified.

Furthermore, in the above-mentioned application example, the Huffman codes have been used in the data compression, however universal codes, etc., can also be used. The data compression is not limited to the Huffman codes as long as the data compression method can compress the dictionary data with good efficiency by the compression parameter common to each section. Also, the data being compressed and expanded are not limited to the dictionary data but also include a database data consisting of an index and a data.

Also, in the above-mentioned application example, the file search processing program and the compression file are copied in the auxiliary storage device 3 and searched. However, without copying them in the auxiliary storage device 3, the program and the compressed file are expanded to the main storage device 2, and processing similar to the above-mentioned processing may also be carried out.

Also, with the compression algorithm being used in the above-mentioned application example, the compression rate is improved by the data compression processing of the Huffman codes at ordinary 8 bits, and the area of the compressed file being recorded on the storage medium such as hard disk explained as the CD-ROM and auxiliary storage device is reduced. The compression rate is improved by the compression algorithm, however the processing time for expanding the compressed file is little changed from the ordinary Huffman code compression.

The time required for the search processing consists of a seeking time of the reader (drive), a reading time of the compressed file, and an expansion processing time.

As mentioned above, with the improvement of the algorithm by the compression algorithm, since the recording area of the compressed file being recorded on the storage medium is reduced, the seeking time of the time required for the search /17
processing of the search program is reduced. As a result, the search speed is improved. This effect is further distinct along with the performance improvement of the hardware.

Hereto, the present invention has been explained by the application example, however needless to say, the present invention can be variously modified and improved within the range of the present invention.

Claims /18

1. A file processing method, characterized by consisting of a step that divides a data and an index data for said data

into several sections and attains a compressed file by compressing and a step that stores said compressed file and the address information after the compression of the above-mentioned sections in a storage medium.

2. The file processing method of Claim 1, characterized by the fact that the above-mentioned sections have a fixed length.

3. The file processing method of Claim 1, characterized by the fact that the above-mentioned sections have a variable length; and the above-mentioned storage step also stores an address information before compressing in the above-mentioned storage medium.

4. The file processing method of any of Claims 1-3, characterized by further including a step that reads the above-mentioned compressed file out of the above-mentioned storage medium for each of the above-mentioned sections, expands it, and stores the data and the index data.

5. The file processing method of Claim 4, characterized by further including a step that stores the restored the restored data and index data in an auxiliary storage device.

6. The file processing method of any of Claims 1-5, characterized by the fact that the above-mentioned compression

step uses a compression algorithm and a compression parameter common to the data and the index data for each section.

7. The file processing method of Claim 1, characterized by the fact that at the above-mentioned compression step, a prescribed number of first bit codes of the above-mentioned data is selected in the appearance frequency sequence; the unselected remaining first bit codes are decomposed into second bit codes; a conversion table is prepared based on the selection result of said second bit codes in the appearance frequency sequence; and the data are compressed based on said conversion table.

8. The file processing method of any of Claims 1-7, characterized by the fact that the above-mentioned data is a /19 dictionary data.

9. A file processing method, characterized by consisting of a step that divides a data and an index data for said data into several sections and reads a compressed file, which is attained by compressing, along with the address information after the compression of the above-mentioned sections out of a storage medium for each of the above-mentioned sections and a step that expands said compressed file and restores the data and the index data.

10. The file processing method of Claim 9, characterized by further including a step that stores the restored data and index data in an auxiliary storage device.

11. The file processing method of Claim 9 or 10, characterized by the fact that at the above-mentioned expansion step, a prescribed number of first bit codes of the above-mentioned data is selected in the appearance frequency sequence during the compression; the unselected remaining first bit codes are decomposed into second bit codes; a conversion table is prepared based on the selection result of said second bit codes in the appearance frequency sequence; and the data are expanded based on said conversion table.

12. The file processing method of any of Claims 9-11, characterized by the fact that the above-mentioned sections have a fixed length.

13. The file processing method of any of Claims 9-11, characterized by the fact that the above-mentioned sections have a variable length; and the above-mentioned storage step also stores an address information before compressing in the above-mentioned storage medium.

14. The file processing method of any of Claims 9-13, characterized by the fact that the above-mentioned data is a dictionary data.

15. A file processing device, characterized by being equipped with a means that divides a data and an index data for said data into several sections and attains a compressed file by compressing and a means that stores said compressed file and the address information after the compression of the above-mentioned sections in a storage medium.

16. The file processing device of Claim 15, characterized by the fact that the above-mentioned sections have a fixed length.

17. The file processing device of Claim 15, characterized by the fact that the above-mentioned sections have a variable length; and the above-mentioned storage step also stores an address information before compressing in the above-mentioned storage medium.

18. The file processing device of any of Claims 15-17, characterized by further including a means that reads the above-mentioned compressed file out of the above-mentioned storage medium for each of the above-mentioned sections, expands it, and stores the data and the index data.

19. The file processing device of Claim 18, characterized by further including a means that stores the restored the restored data and index data in an auxiliary storage device.

20. The file processing device of any of Claims 15-19, characterized by the fact that the above-mentioned compression means uses a compression algorithm and a compression parameter common to the data and the index data for each section.

21. The file processing device of Claim 15, characterized by the fact that in the above-mentioned compression means, a prescribed number of first bit codes of the above-mentioned data is selected in the appearance frequency sequence; the unselected remaining first bit codes are decomposed into second bit codes; a conversion table is prepared based on the selection result of said second bit codes in the appearance frequency sequence; and the data are compressed based on said conversion table.

22. The file processing device of any of Claims 15-21, characterized by the fact that the above-mentioned data is a dictionary data.

23. A file processing device, characterized by consisting of a means that divides a data and an index data for said data into several sections and reads a compressed file, which is attained by compressing, along with the address information after the compression of the above-mentioned sections out of a storage medium for each of the above-mentioned sections and a means that expands said compressed file and restores the data and the index data.

24. The file processing device of Claim 23, characterized by further including a means that stores the restored data and index data in an auxiliary storage device. /21

25. The file processing device of Claim 23 or 24, characterized by the fact that in the above-mentioned expansion means, a prescribed number of first bit codes of the above-mentioned data is selected in the appearance frequency sequence during the compression; the unselected remaining first bit codes are decomposed into second bit codes; a conversion table is prepared based on the selection result of said second bit codes in the appearance frequency sequence; and the data are expanded based on said conversion table.

26. The file processing device of any of Claims 23-25, characterized by the fact that the above-mentioned sections have a fixed length.

27. The file processing device of any of Claims 23-25, characterized by the fact that the above-mentioned sections have a variable length; and the above-mentioned storage step also stores an address information before compressing in the above-mentioned storage medium.

28. The file processing device of any of Claims 23-27, characterized by the fact that the above-mentioned data is a dictionary data.

29. A storage medium, characterized by the fact that in a storage medium that stores the information readable by a computer, it stores a program, which is equipped with a means that divides a data and an index data for said data into several sections and reads a compressed file, which is attained by compressing, along with the address information after the compression of the above-mentioned sections out of the above-mentioned storage medium for each of the above-mentioned sections and a means that expands said compressed file and restores the data and the index data, in the computer.

30. The storage medium of Claim 29, characterized by the fact that a means for storing the restored data and index data in an auxiliary storage device is further included in the computer.

31. The storage medium of Claim 29 or 30, characterized by the fact that the above-mentioned sections have a fixed length.

32. The storage medium of Claim 29 or 30, characterized by the fact that the above-mentioned sections have a variable /22 length; and the above-mentioned reading means also reads an address information before compressing out of the above-mentioned storage medium.

33. The storage medium of any of Claims 29-32, characterized by the fact that the above-mentioned compressed file is attained by compressing using a compression algorithm and a compression parameter common to the data and the index data for each section.

34. The storage medium of any of Claim 29-33, characterized by the fact that the above-mentioned data is a dictionary data.

35. A storage medium, characterized by the fact that in a storage medium that stores the information readable by a computer, a data and an index data for said data are divided into several sections; a compressed file attained by compressing is stored along with the address information after the compression of the above-mentioned sections in a storage medium; and said compressed file is attained by compressing using a compression algorithm and a compression parameter common to the data and the index data for each section.

36. The storage medium of Claim 35, characterized by the fact that the above-mentioned sections have a fixed length.

37. The storage medium of Claim 35, characterized by the fact that the above-mentioned sections have a variable length; and an address information before compressing is also stored.

38. The storage medium of any of Claim 35-37, characterized by the fact that the above-mentioned data is a dictionary data.

39. A storage medium, characterized by the fact that in a storage medium that stores the information readable by a computer, it stores a program, which implements a sequence that divides a dictionary data and an index data for said dictionary data into several sections and attains a compressed dictionary file by compressing and a sequence that stores said compressed dictionary file and stores the storage medium, in the /23
computer.

40. The storage medium of Claim 39, characterized by the fact that the above-mentioned sections have a fixed length.

41. The storage medium of Claim 39, characterized by the fact that the above-mentioned sections have a variable length; and the above-mentioned reading means also reads an address information before compressing out of the above-mentioned storage medium.

42. The storage medium of any of Claims 39-41, characterized by the fact that the above-mentioned compressed dictionary file is attained by compressing using a compression algorithm and a compression parameter common to the data and the index data for each section.

43. The storage medium of any of Claims 39-42, characterized by the fact that a program for implementing a sequence that reads the above-mentioned compressed dictionary file out of the above-mentioned storage medium for each of the above-mentioned sections, expands it, and restores the dictionary data and the index data is further recorded in the computer.

44. The storage medium of Claim 43, characterized by the fact that a program for implementing a sequence that stores the restored dictionary data and index data in the auxiliary storage device is further recorded in the computer.

45. The storage medium of Claim 39, characterized by the fact that a program for implementing a sequence that selects a prescribed number of first bit codes of the above-mentioned data in the appearance frequency sequence, decomposes the unselected remaining first bit codes into second bit codes, prepares a conversion table based on the selection result of said second bit codes in the appearance frequency sequence, and compresses the data based on said conversion table is further recorded in the computer.

46. A storage medium that can read a compressed file by a computer, characterized by the fact that in a storage medium that can read a compressed file by a computer, it has a

compressed data area where a data and an index data for said data are divided into several sections and a compressed data is recorded, a storage address information area where the address information after the compression of the above-mentioned /24 section is stored, and a compression parameter area where the compression parameter used in the above-mentioned compression processing is stored.

47. The storage medium of Claim 46, characterized by the fact that the above-mentioned compression parameter is a prescribed number of first bit codes selected in the appearance frequency sequence of the above-mentioned dictionary data and its appearance frequency and second bit codes selected in the appearance frequency sequence from the second bit codes, in which said unselected remaining first bit codes are divided, and its appearance frequency.

// Insert Figures 1-7 //

Figure 1:

2	Main storage device
3	Auxiliary storage device
4	Input device
5	Display device
6	Input and output device

Figure 2:

1. Compression parameter arithmetic processing

2. End

S1 A dictionary file is opened.

S2 One character (16-bit code) is read out.

S3 Addition of an appearance frequency counter of said code

S4 Has the final character been processed?

S5 The dictionary file is closed.

S6 16-bit codes are sorted in the appearance frequency sequence.

S7 selection of 1024 pieces of codes in the appearance frequency sequence

S8 The rest is decomposed into 8-bit codes, and the appearance frequency is calculated.

S9 The appearance frequency of 8-bit codes is corrected.

S10 A save file of the compression parameter is opened.

S11. 1024 pieces of 16-bit codes and the appearance frequency are written.

S12 The appearance frequency of 256 pieces of 8-bit codes is written.

S13 The save file of the compression parameter is closed.

Figure 3:

1. 16-bit codes

2. 8-bit codes

3. (2 + 2) bytes x i kinds (for example, 1024)

   2 bytes x 256 kinds

4. Symbol 1 code

5. Appearance frequency

6. Symbol 2 code

7. Symbol i code

Figure 4:

1. Data compression processing and address information arithmetic processing

2. End

S21 A Huffman tree is prepared from the appearance frequency of 8 and 16-bit codes

S22 A dictionary file is opened.

S23 Each save file of the compressed data and the address information is opened.

S24 One section is read out of the dictionary file.

S25 The compressed data of one section is calculated from the Huffman tree.

S26 A finishing code is added.

S27 The compressed data is written into the save file.

S28 The address information is calculated.

S29 The address information is written into the compressed file.

S30 Has the processing section been finished?

S31 Each save file of the compressed data and the address information is closed.

S32 The dictionary file is closed.

Figure 5:

1. Compressed file synthesis processing and compressed file storage processing
2. End

S41 A compression file is opened.

S42 A save file of the compressed data is opened.

S43 The compression parameter is copied in the compressed file.

S44 The save file of the compression parameter is closed.

S45 A save file of the address information is opened.

S46 The address information is copied in the compressed file.

S47 The save file of the address information is closed.

S48 A save file of the compressed data is opened.

S49 The compressed data is copied in the compressed file.

S50 The save file of the compressed data is closed.

S51 Compressed file storage

S52 The compressed file is closed.

Figure 6:

(a) Parameter arithmetic

Compression parameter

(b) Dictionary parameter

(c) Compression processing

Compressed data

(d) Address arithmetic

Address information

(e) Synthesis

Compressed file

1. Huffman codes

Compression parameter

2. First section

3. Second section

4. Final section

5. 1st address

6. 2nd address

7. Last address

8. Management information

9. Compression parameter area

10. Storage address information area

11. Compressed data area

Figure 7:

1. Index reading processing

2. End

S61 The address of the highest index is set.

S62 ||Call of expansion processing (highest index)||

S63 The address of the upper index of the initial letter of a keyword is acquired.

S64 ||Call of expansion processing (highest index)||

S65 The address of the lower index of the next hierarchy in the keyword is acquired.

S66 ||Call of expansion processing (lower index of the next hierarchy)||

S67 Is the expansion of the highest index finished?

FIG. 1

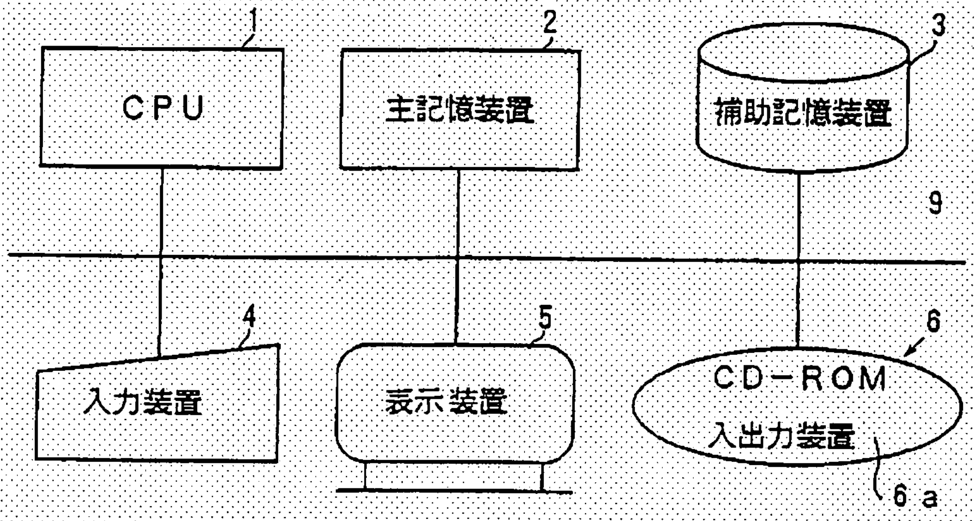

1
CPU
2
主記憶装置
3
補助記憶装置
9
4
入力装置
5
表示装置
6
CD－ROM
入出力装置
6 a

FIG. 2

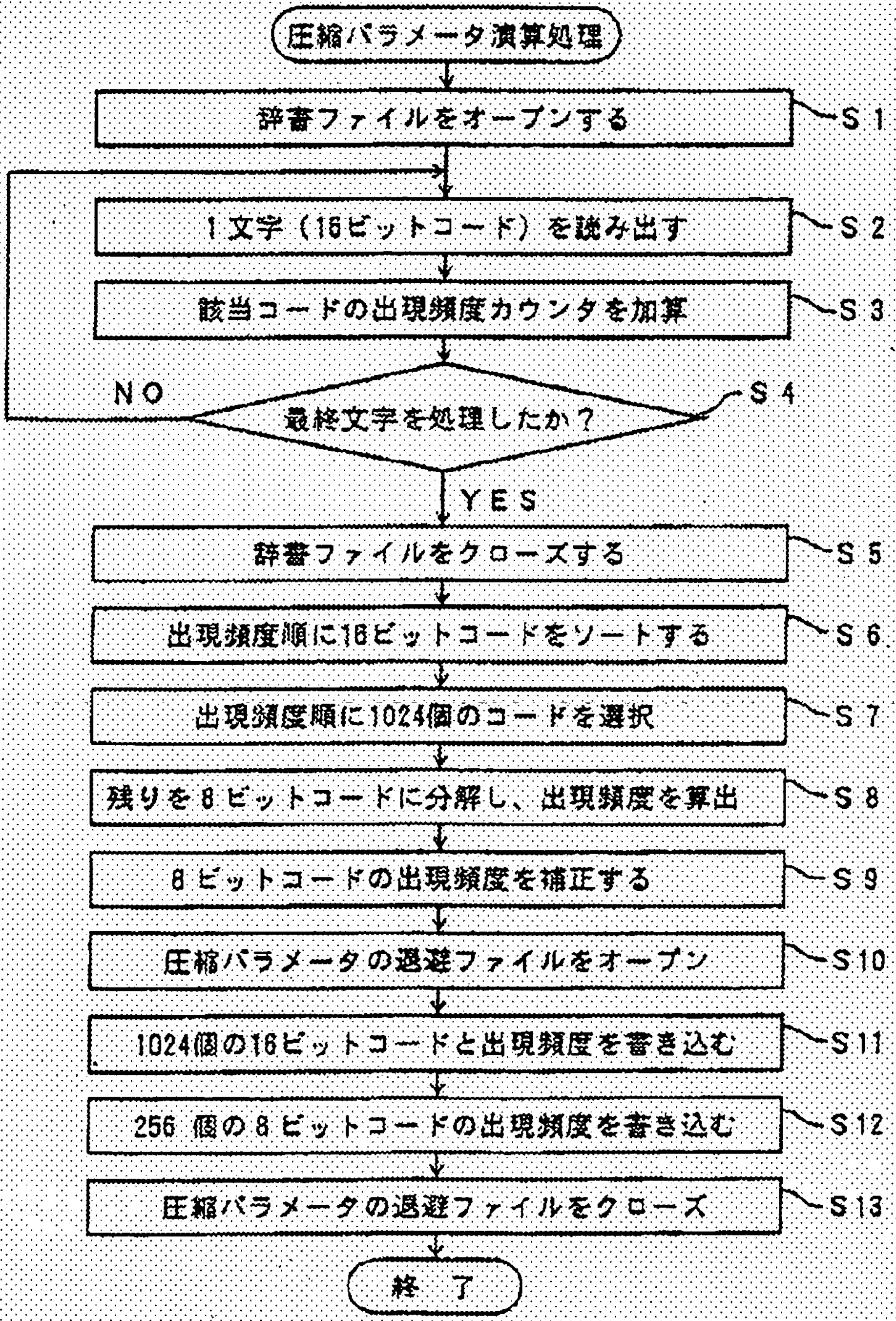

圧縮パラメータ演算処理
辞書ファイルをオープンする
S1
1文字（16ビットコード）を読み出す
S2
該当コードの出現頻度カウンタを加算
S3
最終文字を処理したか？
S4
NO
YES
辞書ファイルをクローズする
S5
出現頻度順に16ビットコードをソートする
S6
出現頻度順に1024個のコードを選択
S7
残りを8ビットコードに分解し、出現頻度を算出
S8
8ビットコードの出現頻度を補正する
S9
圧縮パラメータの退避ファイルをオープン
S10
1024個の16ビットコードと出現頻度を書き込む
S11
256個の8ビットコードの出現頻度を書き込む
S12
圧縮パラメータの退避ファイルをクローズ
S13
終了

FIG. 3

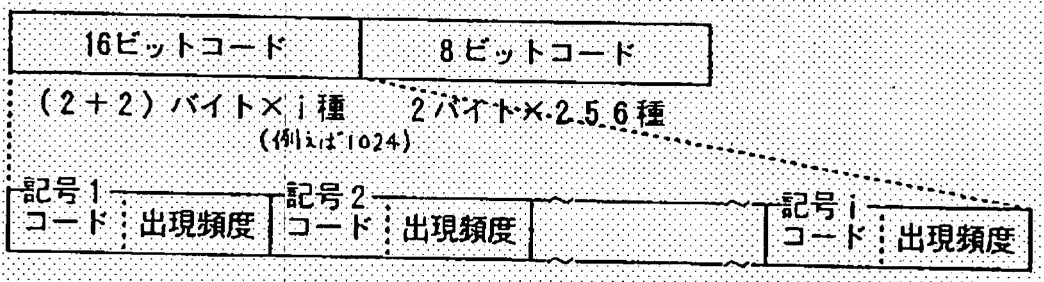
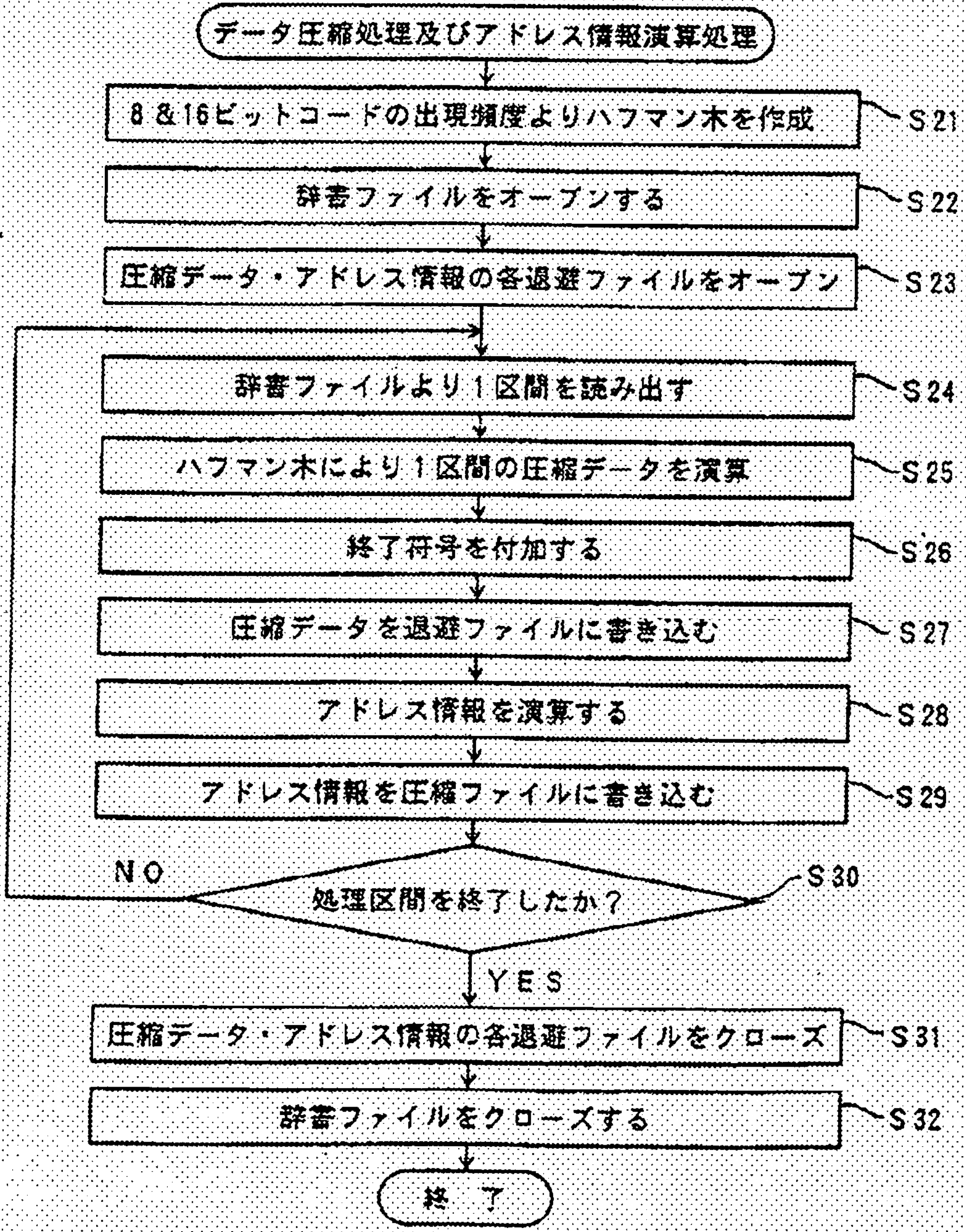

16ビットコード
8ビットコード
（2＋2）バイト×i種
（例えば1024）
2バイト×256種
記号1コード
出現頻度
記号2コード
出現頻度
記号iコード
出現頻度

FIG. 4



データ圧縮処理及びアドレス情報演算処理
8＆16ビットコードの出現頻度よりハフマン木を作成
S21
辞書ファイルをオープンする
S22
圧縮データ・アドレス情報の各退避ファイルをオープン
S23
辞書ファイルより1区間を読み出す
S24
ハフマン木により1区間の圧縮データを演算
S25
終了符号を付加する
S26
圧縮データを退避ファイルに書き込む
S27
アドレス情報を演算する
S28
アドレス情報を圧縮ファイルに書き込む
S29
NO
処理区間を終了したか？
S30
YES
圧縮データ・アドレス情報の各退避ファイルをクローズ
S31
辞書ファイルをクローズする
S32
終　了

FIG. 5

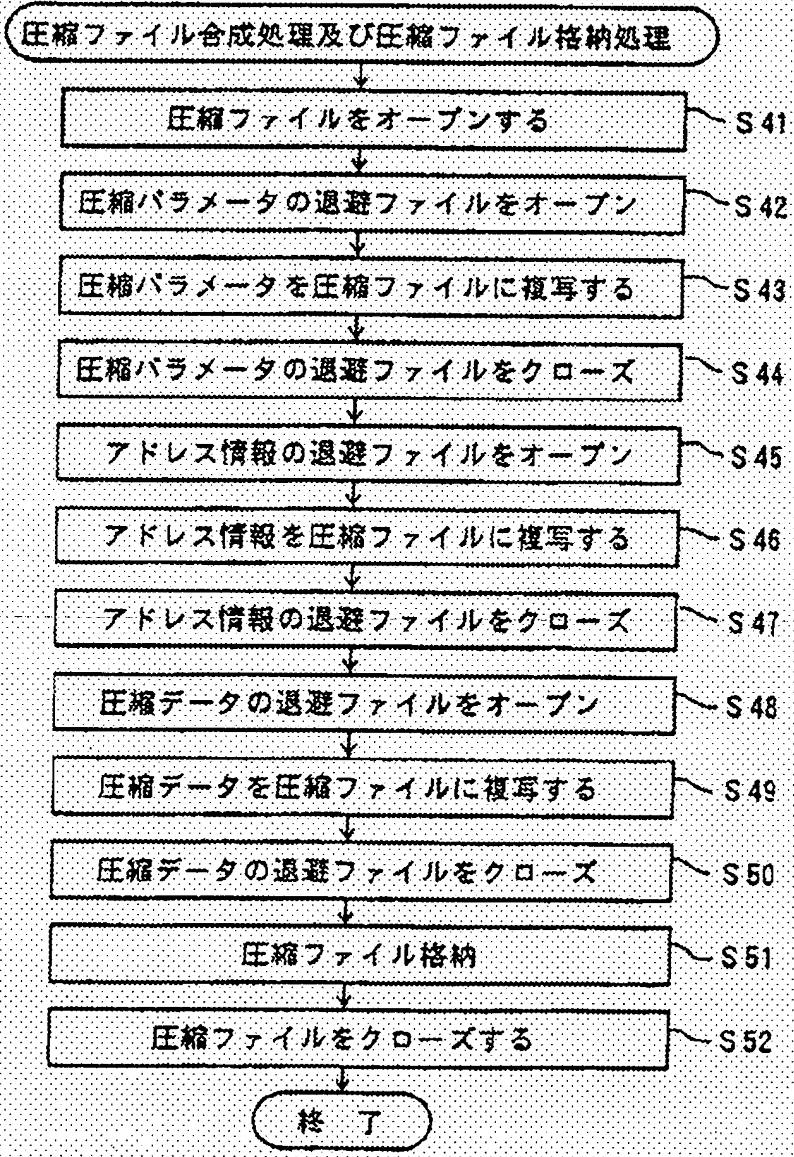

圧縮ファイル合成処理及び圧縮ファイル格納処理
圧縮ファイルをオープンする
S41
圧縮パラメータの退避ファイルをオープン
S42
圧縮パラメータを圧縮ファイルに複写する
S43
圧縮パラメータの退避ファイルをクローズ
S44
アドレス情報の退避ファイルをオープン
S45
アドレス情報を圧縮ファイルに複写する
S46
アドレス情報の退避ファイルをクローズ
S47
圧縮データの退避ファイルをオープン
S48
圧縮データを圧縮ファイルに複写する
S49
圧縮データの退避ファイルをクローズ
S50
圧縮ファイル格納
S51
圧縮ファイルをクローズする
S52
終　了

FIG. 6

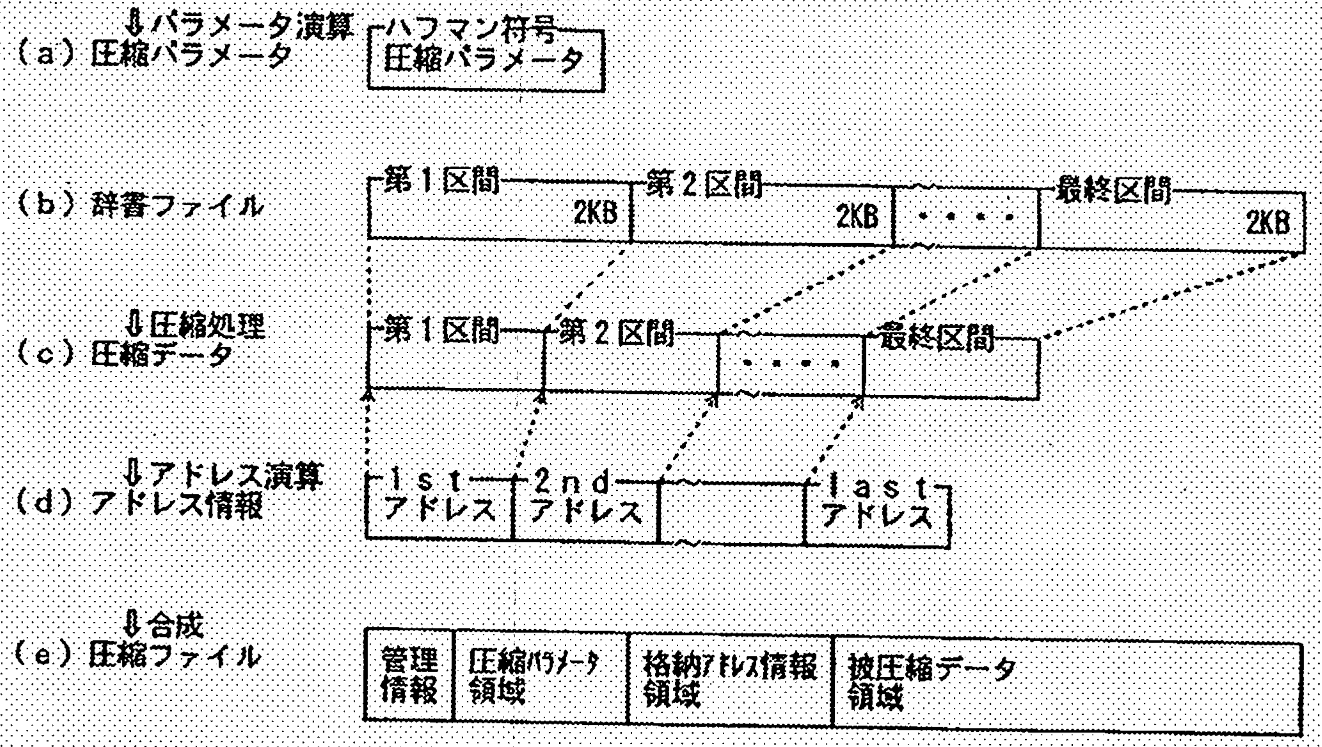
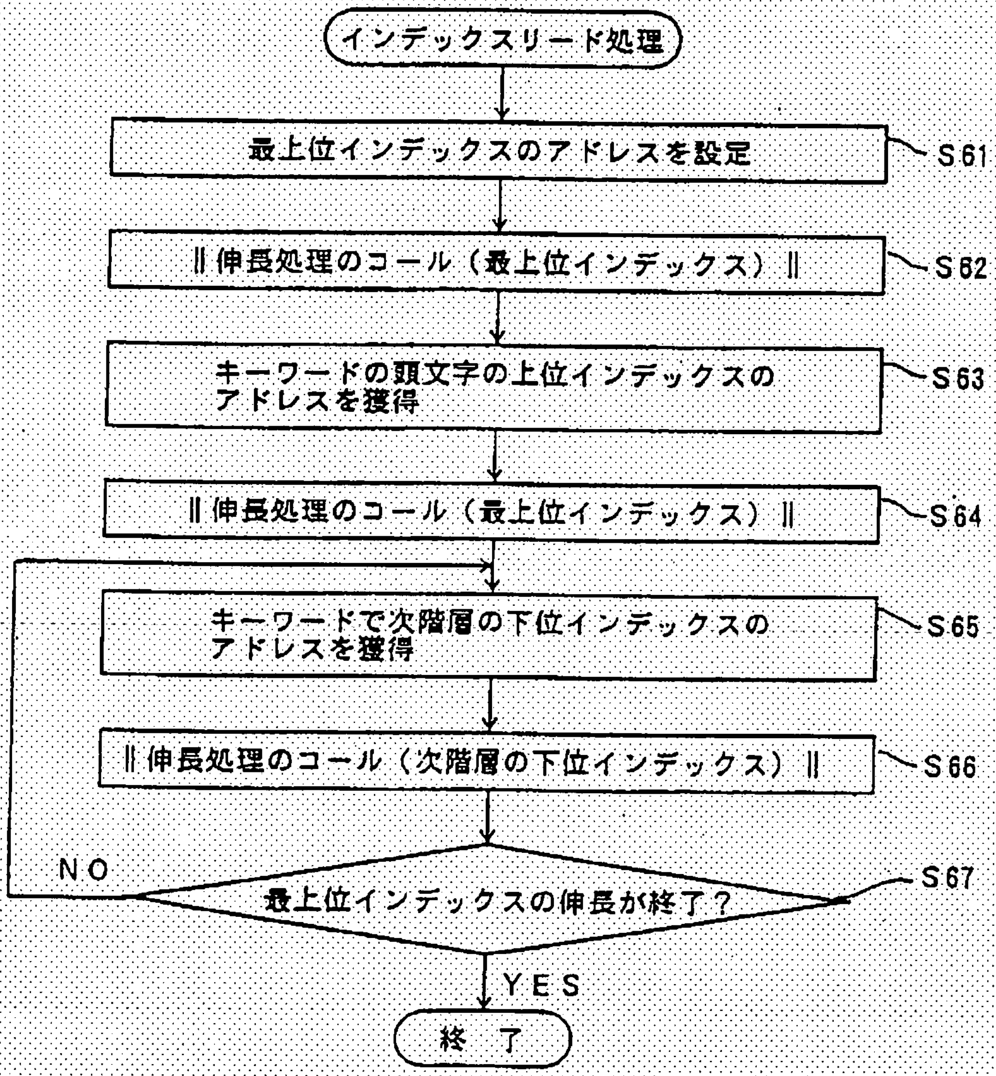

⇩パラメータ演算
(a) 圧縮パラメータ
ハフマン符号
圧縮パラメータ
(b) 辞書ファイル
第1区間
2KB
第2区間
2KB
最終区間
2KB
⇩圧縮処理
(c) 圧縮データ
第1区間
第2区間
最終区間
⇩アドレス演算
(d) アドレス情報
1st アドレス
2nd アドレス
last アドレス
⇩合成
(e) 圧縮ファイル
管理情報
圧縮パラメータ領域
格納アドレス情報領域
被圧縮データ領域

FIG. 7


インデックスリード処理
最上位インデックスのアドレスを設定
S61
‖伸長処理のコール（最上位インデックス）‖
S62
キーワードの頭文字の上位インデックスのアドレスを獲得
S63
‖伸長処理のコール（最上位インデックス）‖
S64
キーワードで次階層の下位インデックスのアドレスを獲得
S65
‖伸長処理のコール（次階層の下位インデックス）‖
S66
NO
最上位インデックスの伸長が終了？
S67
YES
終了